Formal Module Systems and Nuprl-Light
A Programmer's Perspective

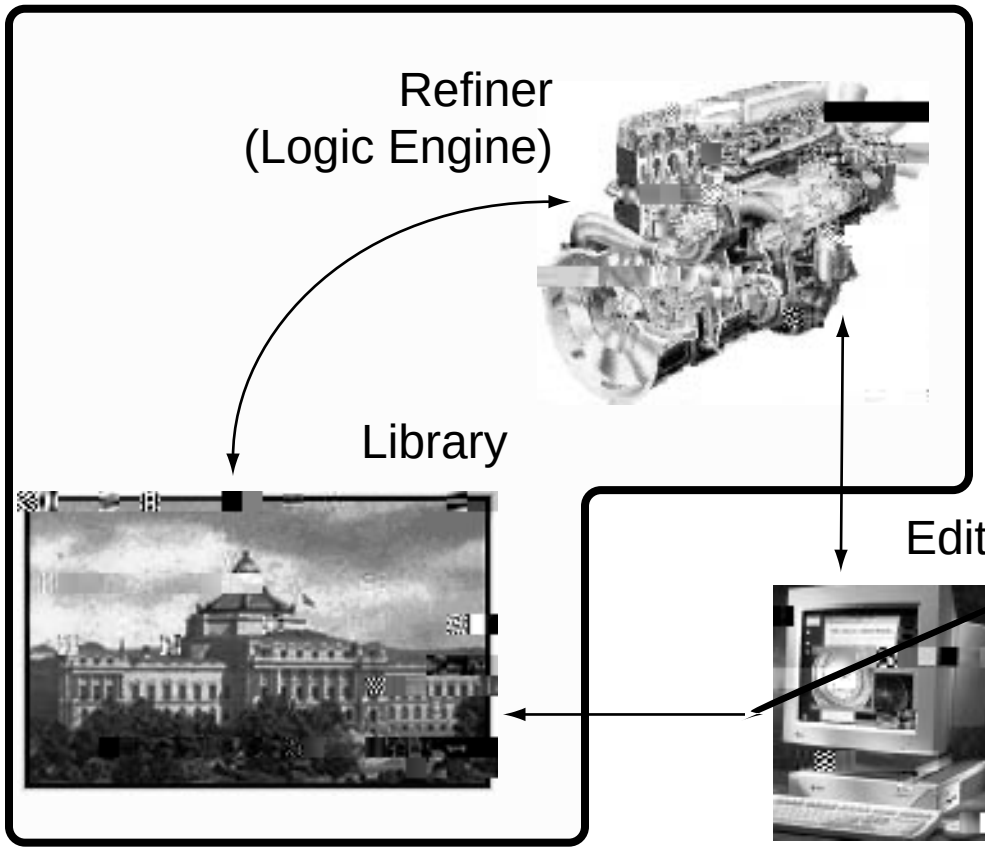
Jason Hickey
Cornell University

- Review of formal modules
- The formal module system
- Additional informal components
- The ML module system
- Examples

Formal Modules

- A theory is a collection of:
  - axioms (statements in a theory)
  - theorems
- Relyormation, propositions-as-types
  - axioms are assumptions
  - theorems are implementations
- Each theory has a formal signature
  - dependent record of axiom types
  - implementation is a record
- Object-oriented features (inheritance, subtyping, etc)

Overall Architecture



Movable Point

theory *Point* (*ITT*) = *f* axiom *x*: *Zg*

theory *MovablePoint* (*Point*) =
axiom *mv* *x*: *T* ! *T*

axiom *8T ‘ Point*: *8*

T Set with decidable membership

theory $Set_1^0\ (T{:}U_i)fITTg = f$
axiom $member{:}T\ !\ P_i$
g

theory $Set\ \ (T{:}Ui)fSet$

$i$axiom $:car\ !\ T\ !\ P$

axiom $union{:}car\ !\ car\ !$
axiom $8s_{\}};s\,2\ \ :8\ :T{:}\ member(;s$

theory *LeftMonoid f ITT g = f*

axiom *car*: U_i

axiom *'*: *car ! car ! car*

axiom *·*: *car ! car ! P_i*

axiom 1: *car*

axiom *8m*: *car*: *m ' 1 · m*

$$$

g

theory *Group f g = f*

axiom *8m*: *ca : 9n*: *car*: *m ' n · 1*

theorem *8m*: *car*: *n ' m · 1*

g

General Features

- Theories have dependencies (Group : Monoid)
- All properties of parents are inherited by children
- Inhabitants are formed from proofs

  Coercion A $\rightarrow$

System support

Module system

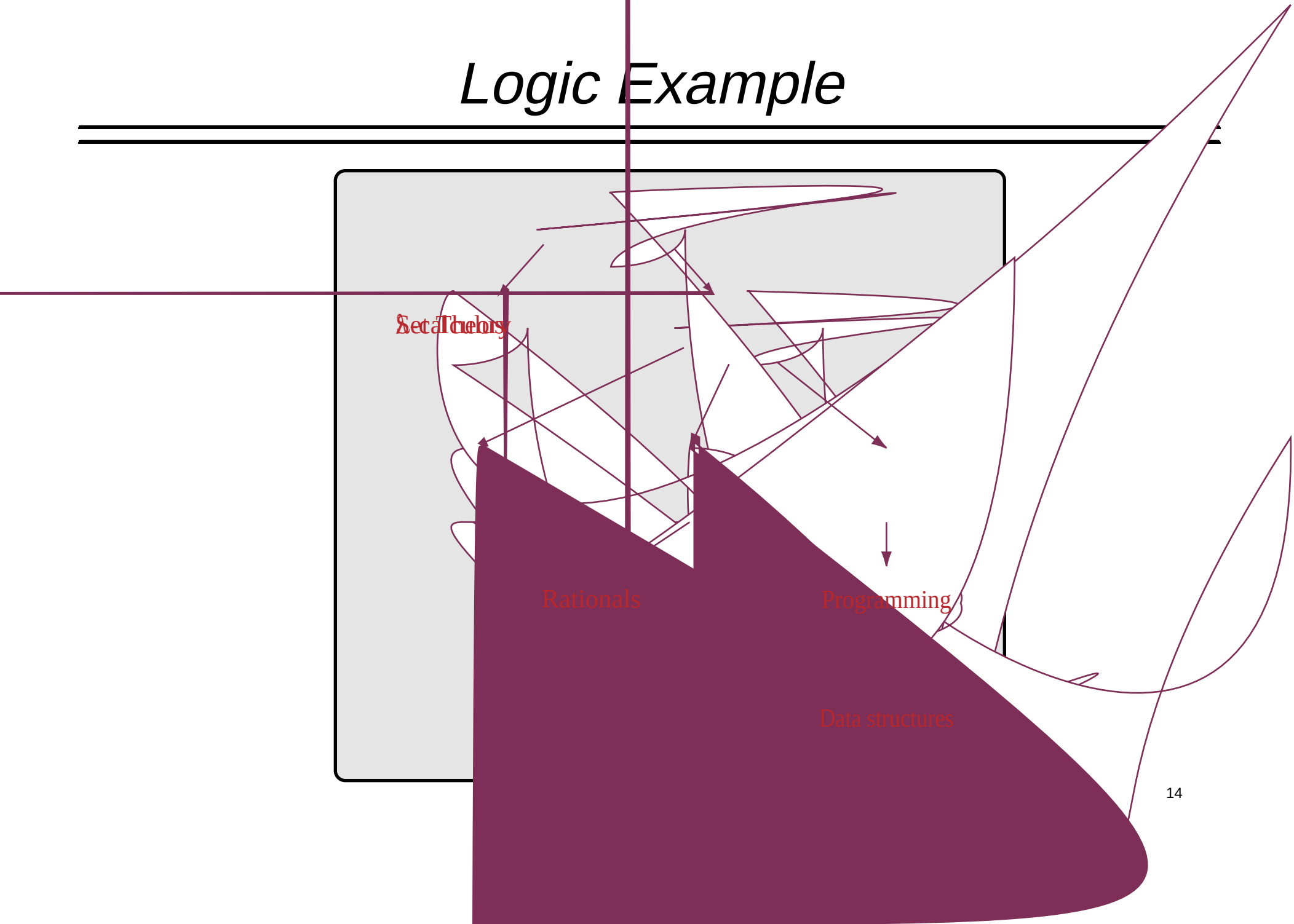
- Theories also contain “informalS24 objects:
  - Inference rules
  - Theorem proving tactics
  - Display forms
- None of these objects have types
- Any reasonable module system must manage informal objects

Nuprl-Light

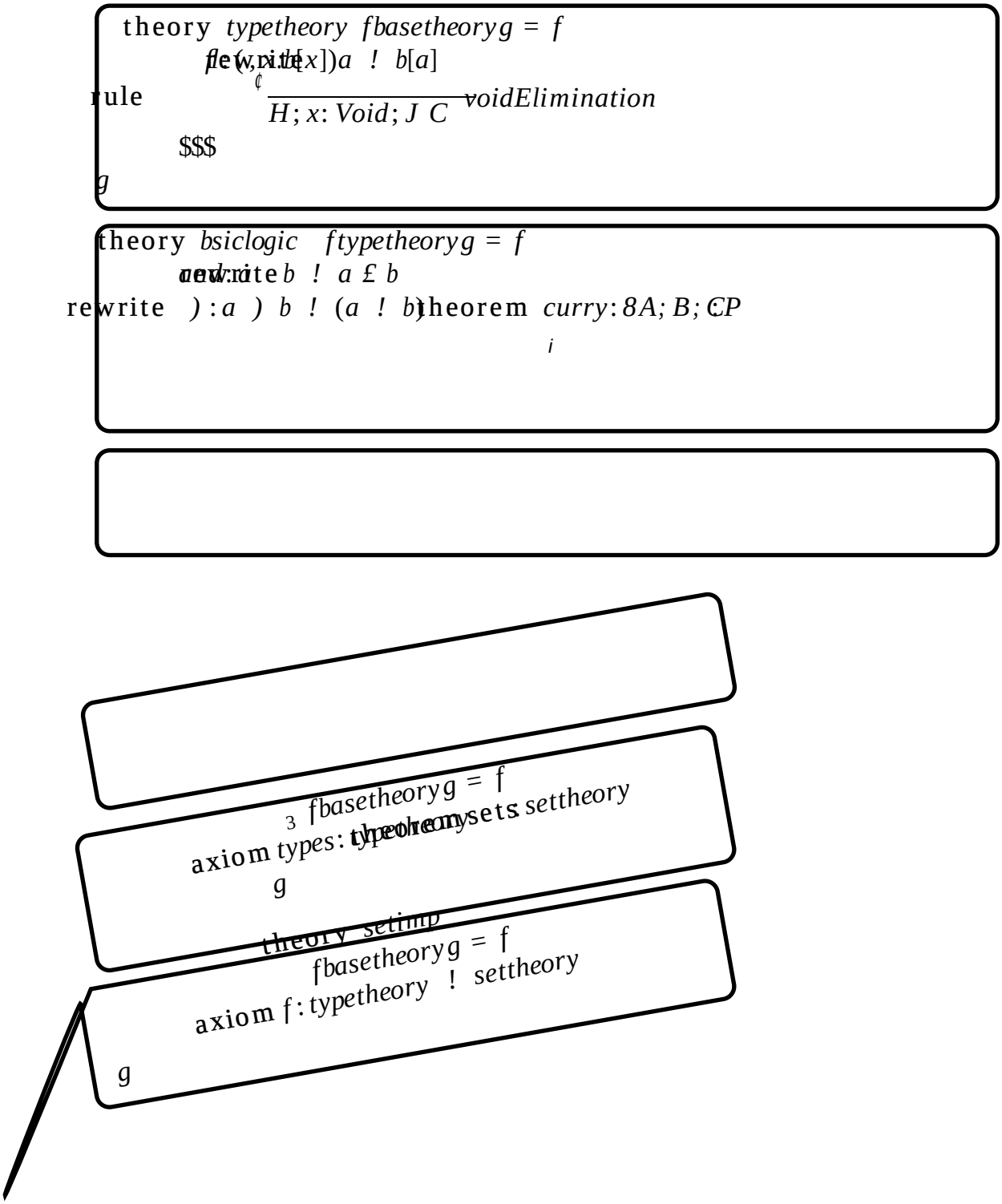
System goals

- Modularize:
  - the type theory
  - the tactics
- Allow multiple mathematical domains
- Allow separate proof development
- Interactive use is secondary at the moment
  - Adopt HOL-style interface for now
  - Wait for a term editor
- Preserve legacy proofs
- Nuprl 5 plugin

Logic Example



Theory design


theory typetheory fbasetheoryg = f
rule
$$$
g
theory bsiclogic ftypetheoryg = f
rewrite
rewrite
axiom types: typetheory sets: settheory
g
theory setimp
fbasetheoryg = f
axiom f: typetheory ! settheory
g

Theory objects

- Items
  - axiom <name> : <type>
  - rewrite <name> : <redex> -> <contractum>
  - rule <name> : <inference> <side-condition>
  - theorem <name> proves <axiom-name | rule-name | rw-name>
  - display-form <name> for <term>

- theory <name> (params) {[parents...] }= {[items...] }

Differences from normal module systems

- Object oriented (inheritance + abstraction)
- Signature is generated from axioms/rules
- Theorems provide default values

Implementation in Caml-Special-Light

- CSL module system is more rigid

  (Theory will be one or more CSL modules)

  - 4 0 412.344 621.5 Tmı˝(·)Tjı˝-0.5 -2.063 TDı˝0.007 Tcı˝-0.014 Twı˝[(F

Theory implementation

- theory <name> : {[parents...] }= {[values...] }

- module type <name> (parents) =
  sig
      [value declarations]
  end

- module <name> (parents) =
  struct
      let refiner' = ref (join_refiners parents)
      [valu definitions]
      let refiner = !refiner'
  end

Implementation

- A rule:

  S •> G
  by <name> [params...] [side-condition]
  1. S_1 •> G_1
  ...
  n. S_n •> G_n

7267

Implementation

- A rewrite
  - rewrite <name> : <redex> -> <contractum|
  - val <name> : <evaluator>
  - let <nam|e> = create_rewrite refiner' <name> <redex> <contractum
- A theorem for an axiom or rule:
  - theorem%£>es <axiomam|e | rule-name>
  - val <name> : <tactic>
  - let <namreate_theorem refiner' <nam¤Ac2Z3àó•
  - These are usually created interactively

Operations on refiners

Notes

tactics and evaluators are always correct (the return the

Status

- Batch mode refiner
  - Rules/tactics are being ported
  - Parallel withod's refinr
  - Moleem is moly complete
  - Non-sddard modules?
  - Btch mode refinement is attractive
- Interactive proof development
  - HOL-style proofinemend is easy (just top loop)
  - Emacs is an easy hack
  - Add Nuprl 5 editor + library
  - Synthesizer generator editor